MORL-Prompt: An Empirical Analysis of Multi-Objective Reinforcement Learning for Discrete Prompt Optimization

Yasaman Jafari **Dheeraj Mekala** **Rose Yu** **Taylor Berg-Kirkpatrick**

University of California San Diego

{yajafari, dmekala, roseyu, tberg}@ucsd.edu

Abstract

RL-based techniques can be employed to search for prompts that, when fed into a target language model, maximize a set of user-specified reward functions. However, in many target applications, the natural reward functions are in tension with one another – for example, content preservation vs. style matching in style transfer tasks. Current techniques focus on maximizing the average of reward functions, which does not necessarily lead to prompts that achieve *balance across rewards* – an issue that has been well-studied in the multi-objective and robust optimization literature. In this paper we conduct an empirical comparison of several existing multi-objective optimization techniques, adapted to this new setting: RL-based discrete prompt optimization. We compare two methods optimizing the volume of the Pareto reward surface, and one method that chooses an update direction that benefits all rewards simultaneously. We evaluate performance on two NLP tasks: style transfer and machine translation, each using three competing reward functions. Our experiments demonstrate that multi-objective methods that directly optimize the volume of the Pareto reward surface perform better and achieve a better balance of all rewards than those that attempt to find monotonic update directions.

1 Introduction

Discrete prompt tuning involves refining a text prompt for a language model (LM) to maximize a set of user-specified objectives on the LM's output (Shin et al., 2020; Schick and Schütze, 2020; Wen et al., 2023). Successful techniques for prompt tuning allow users to control and adapt powerful LLMs to new tasks without the trial-and-error of manual prompt design. While RL-based techniques have been shown to be effective at finding prompts that optimize an average of rewards (Deng et al., 2022), in many target applications, there is a tension between the natural reward functions.

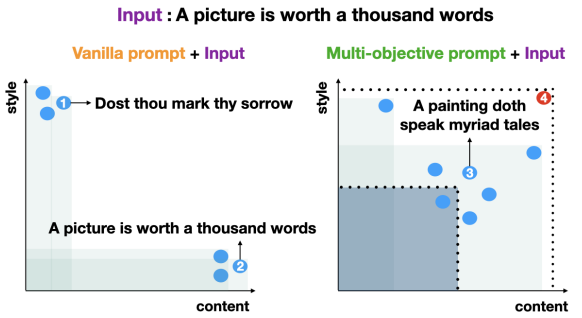


Figure 1: A modern to Shakespearean text style transfer setting where each dot represents an output sentence sampled from an LM conditioned on either a prompt trained with average reward (left) or a prompt trained using multi-objective optimziation techniques (right). The output sample 1 only optimizes for style match, while output sample 2 only addresses content preservation. Sample 3, on the other hand, balances both objectives at the same time. The shaded regions indicate measures of volume of the Pareto reward surface.

For example, as depicted in Figure 1, many style transfer tasks need to preserve content while simultaneously maximizing transfer into the target style – two objectives that are directly at odds with one another. Thus, current techniques result in a phenomenon we will refer to as *objective collapse*: focusing on maximizing the average of reward functions (also called *scalarization*) can lead to prompts that disproportionately maximize a subset of objectives at the expense of others. For instance, in Figure 1, the prompt on the left side tends to produce LM outputs (represented by blue dots) that prioritize one objective over the other. Conversely, the prompt on the right side produces samples that achieve reasonable performance across all objectives simultaneously. However, in both cases the *average reward* is nearly equivalent.

The problem of reward balancing has been well-studied in other domains—for example, the multi-objective and robust optimization literature proposes several approaches that offer advantages

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 9878–9889

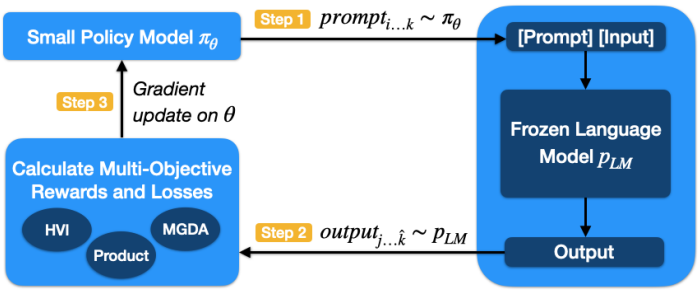


Figure 2: In all the settings, we have a parameter-efficient policy model, responsible for generating the task-specific prompts, where all the parameters of the model except for an MLP module are frozen. Another frozen language model is used to generate output sentences, given an input and a prompt from the policy model. All the output sentences are then evaluated with respect to each objective, and multi-objective losses are calculated. Finally, a gradient update on the MLP parameters is performed.

over scalarization. However, these techniques have not been applied to the RL-based discrete prompt optimization setting that is most relevant in NLP. Thus, in this paper we conduct an empirical comparison of several existing techniques for multi-objective optimization that we adapt to discrete prompt optimization, where we aim to evaluate their effectiveness in achieving a more useful balance of rewards in downstream prompt-driven NLP tasks. The first two approaches we compare maximize the volume of the Pareto reward surface, while the third method chooses a gradient update direction that is beneficial for all rewards simultaneously.

More specifically, the first method in our study computes the hypervolume indicator (HVI) (Knowles et al., 2004) for a set of samples drawn from a given prompt, and treats this measure as the final reward in RL. Intuitively, HVI measures the area under the Pareto frontier of the outputs sampled from the current prompt (shown by the shaded regions in Figure 1). Samples that achieve a better balance of reward elevate the Pareto frontier and increase HVI. However, this method has a potential drawback: if an outlier sample (e.g., represented by the red dot labeled with a four in Figure 1) achieves high values across all rewards, the HVI can be disproportionately high (represented by the outer rectangular region in Figure 1, which dominates the shaded areas). This *dominant outlier effect* may diminish the stability of HVI optimization in an RL setting, as it becomes very sensitive to outliers.

Therefore, we also investigate using a simpler method for maximizing the volume in the second approach, called the expected product of rewards. Here, we approximate the expected volume by simply computing the average *product of rewards* (tentatively depicted by the dark rectangular region in Figure 1).

The third approach takes a different strategy based on *steepest gradient descent* (Fliege and Svaiter, 2000). We approximate the gradient of the expectation of each individual reward separately and then search for an update direction to make monotonic progress in every reward simultaneously.

To understand the effectiveness of these approaches in the discrete prompt optimization setting, we conduct experiments on two text generation tasks: text style transfer and machine translation using three competing reward functions for each task. Our findings indicate that volume-based methods are most effective in this setting, achieving substantial gains in balancing the competing rewards, compared to the baseline methods. While RL-based steepest descent also improves balance, it does not perform as robustly as the volume-based methods.

2 Problem Statement

In this paper, we specifically focus on optimizing discrete prompts, as they offer the advantages of interpretability and reusability in contrast to continuous or "soft" prompts. We acknowledge that the issue of balancing multiple conflicting objectives is a well-established area of research within the multi-objective and robust optimization literature. However, adapting these techniques to the domain of discrete prompt optimization for language models comes with challenges due to having sources of discontinuity and discreteness. First, the text tokens for the prompt are discrete, and second, since marginalizing over

Algorithm 1: Volume-based policy update for one input sentence.

Input: Input sentence x, policy π_θ, reward models $r_{1...m}$, external frozen LM
$\{z_{1...k}\} \sim \pi_\theta(x)$ ▷ Sample k prompts from the policy
for i = 1. . . k **do**:
    $\{y_{1...\hat{k}}\} \sim p_{LM}(y|x, z_i)$ ▷ Sample $\hat{k}$ output sentences from a desired LM
end for
for i = $1 \ldots k \cdot \hat{k}$ **do**:
    calculate $r_{1...m}(y_i, x)$ ▷ Calculate $r_{1...m}$ for each output sentence y and input x
end for
Calculate r_{prod} (or r_{hvi}) ▷ Calculate expected product of rewards (or hypervolume)
Calculate $\mathcal{L}$ using r_{prod} (or r_{hvi}) ▷ Use SQL loss (Guo et al., 2022)
$\theta = \theta - \eta \nabla_\theta \mathcal{L}(\theta)$ ▷ Gradient descent on policy parameters

Figure 3: In this algorithm, k prompts are sampled from the policy model and used alongside an input sentence to generate $\hat{k}$ output samples from an external frozen LM. The desired objective values for each of the sentences are calculated and combined into a single reward value by computing their expected product of rewards or hypervolume indicator, based on the desired approach. Then, the loss is computed based on this reward and used for a gradient update on the policy LM's parameters. For details on the SQL loss and parameter updates, we refer the interested reader to (Guo et al., 2022).

all possible output samples is intractable, we need to approximate the expected gradient of the loss with respect to the sampled sentences.

We train a small, parameter-efficient policy network in order to generate task-specific prompts that can later be used alongside an input sentence to be fed into any other language model, as depicted in Figure 2. We particularly put an emphasis on optimizing prompts to achieve a balance across multiple reward functions. Given m multiple objectives and their corresponding reward functions $\{r_1, r_2, \ldots r_m\}$, we perform discrete prompt optimization for controlled text generation. We refer to the prompt as z, the input text as x, and the text generated by the LM as y. We aim to generate a prompt that is added to the beginning of the input and causes the LM to generate output text compliant with the objectives.

2.1 Optimization problem

We formulate discrete prompt optimization as an RL problem, where we train a multi-layer perceptron (MLP) module over a frozen language model as our policy network. A frozen LM head is used after the MLP module to generate the prompts.

The RL-based approach to discrete prompt optimization tries to optimize an intractable objective through stochastic approximation, and a common way of incorporating multiple objectives is to use their sum (scalarization). In Equation 1, we show the intractable objective for scalarization that RL-based methods attempt to optimize.

$$\max_\theta \mathbb{E}_{z\sim\pi_\theta}\left[\mathbb{E}_{y\sim p_{LM}(y|x,z)}\left[\sum_{i=1}^{m} r_i(y,x)\right]\right] \quad (1)$$

Note that Equation 1 involves true expectations, which are intractable to compute; therefore, we approximate the expectations by utilizing samples from the policy and the frozen language model. At each step, given a text input x, we sample k prompts $\{z_1, z_2, \ldots z_k\}$ from the policy π_θ, where θ represents the policy parameters. Subsequently, we utilize another frozen language model p_{LM} to generate $\hat{k}$ output sentences for each pair of input x and prompt z_i. Then, we assess the quality of these outputs using the reward function r_i corresponding to each objective[1].

We will explore RL-based approaches that go beyond simply summing rewards, which may offer better ways to balance multiple objectives.

3 Methodology

In this section, we describe the adapted optimization methods for generating discrete prompts that, when fed into an LM along with the input text, produce outputs that maximize a set of competing reward functions. We compare two

[1]For simplicity, we assume the reward value is solely dependent on the generated text y and the input text x. It can be easily expanded to include prompt z or the reference text, if necessary.

optimization methods that maximize the volume coverage of rewards and one method that finds the gradient update direction which optimizes all rewards simultaneously.

We adopt the soft Q-learning (SQL) reinforcement learning framework introduced by (Guo et al., 2022), which has demonstrated effectiveness in discrete prompt optimization with single reward functions or scalarization (Deng et al., 2022). In line with (Deng et al., 2022), we utilize only the on-policy component of SQL. For clarity and simplicity, we omit certain details of the SQL updates in the pseudo-codes (Figures 3 and 4), focusing instead on the novel multi-objective components. For a comprehensive discussion of SQL, we refer the reader to (Guo et al., 2022).

3.1 RL-based Volume Improvement

In this section, we investigate two approaches that aim to improve the volume coverage of rewards.

3.1.1 Hyper-volume indicator

The hypervolume indicator (Zitzler and Thiele, 1998; Knowles et al., 2004) is defined for a point set $S \subset \mathbb{R}^d$ and a reference point $p_{ref} \in \mathbb{R}^d$. The hypervolume indicator H quantifies the region dominated by S and bounded by p_{ref}. S denotes the set of points/solutions that we are examining. Mathematically, hyper-volume indicator is defined as:

$$H(\mathrm{S}) = \Lambda\Big(\Big\{q \in \mathbb{R}^d \mid \exists p \in \mathrm{S} : q \leq p \text{ and } p_{ref} \leq q\Big\}\Big)$$

where the notation q $\leq$ p means that each component of the vector q is less than or equal to each of the corresponding components of the vector p, and $\Lambda(\cdot)$ shows the Lebesgue measure for the sub-space. In other words, $\Lambda(\cdot)$ measures the size of the hypervolume covered by a set of solutions in the objective space. This hypervolume is always measured with respect to a reference point, which we consider to be a zero vector in all our experiments.

In our setting, each point in S is a sampled sentence. For example, in the style-transfer task, if we have 2 objective values of style-match: 0.6 and content-match: 0.3 for a sentence, this point can be denoted as (0.6, 0.3), and the reference point would be set to (0, 0). We consider the hypervolume indicator of the reward functions as the ultimate reward signal for training the policy network in the first approach.

3.1.2 Expected product of rewards

In this method, we consider the expected product of objective functions as the reward signal for training the policy network. We obtain $\hat{k}$ samples as output per prompt and for each sentence, we compute all m reward values, and calculate the product of rewards. We utilize the expected product of rewards across all $\hat{k}$ samples as the final reward signal for policy updates.

The main advantage of this reward compared to the HVI reward is that the effect of the outliers will be more controlled by using the expected value of objectives within a sampled set of sentences.

The pseudo-code for the volume-based approaches is provided in Figure 3, where at each update step, we sample prompts from the policy model and generate output sentences from a desired language model. We then calculate the reward values for each of the objectives separately and use them to compute the dominated hypervolume or the expected product of rewards and use it to calculate the loss. Then, we update the policy model using gradient descent.

3.2 Multiple Gradient Descent Algorithm with RL

We describe the multiple gradient descent algorithm (MGDA), which finds the gradient update direction that maximizes all the rewards. This method follows the approach of steepest descent for multi-criteria optimization (Fliege and Svaiter, 2000), where the goal is to find a direction d_t that improves all the objectives by the amount of α_t, at each step t. Here, L_i and θ represent the expected loss corresponding to objective i, and the parameters of the policy model, respectively.

$$\begin{aligned} &(d_t, \alpha_t) = \arg\min_{d \in \mathbb{R}^n, \alpha \in \mathbb{R}} \alpha + \frac{1}{2}\|d\|^2, \\ &\text{s.t. } \nabla \mathcal{L}_i\left(\theta_t\right)^T d \leq \alpha, \quad i = 1, \ldots, m. \end{aligned} \tag{2}$$

The update rule for the parameters θ at time t with the step size η is defined as:

$$\theta_{t+1} = \theta_t - \eta d_t \tag{3}$$

This approach has been used in continuous multi-objective settings (Sener and Koltun, 2019; Lin et al., 2019). However, in our setting, since we optimize for discrete prompts, we compute stochastic gradient approximations by sampling LLM outputs and then use reinforcement learning to estimate the gradient based on the samples. We

Algorithm 2: MGDA-based policy update for one input sentence.

Input: Input sentence x, policy π_θ, reward models $r_{1...m}$, external frozen LM
$\{z_{1...k}\} \sim \pi_\theta(x)$ ▷ Sample k prompts from the policy
for i = 1. . . k **do**:
    $\{y_{1...\hat{k}}\} \sim p_{LM}(y|x, z_i)$ ▷ Sample $\hat{k}$ output sentences from a desired LM
end for
for i = $1 \ldots k \cdot \hat{k}$ **do**:
    calculate $r_{1...m}(y_i, x)$ ▷ Calculate $r_{1...m}$ for each output sentence y and input x
end for
for i = 1. . . m **do**:
    Calculate $\mathcal{L}_m$ using r_m ▷ Use SQL loss (Guo et al., 2022)
end for
$\lambda_1, \ldots, \lambda_m = \text{FrankWolfeSolver}(\nabla_\theta \mathcal{L}_i(\theta))$ ▷ Find the direction using [7]
$\theta = \theta - \eta \sum_{i=1}^{m} \lambda_i \nabla_\theta \mathcal{L}_i(\theta)$ ▷ Gradient descent on policy parameters

Figure 4: In this algorithm, k prompts are sampled from the policy model and used alongside an input sentence to generate $\hat{k}$ output samples from an external frozen LM. The desired objective values for each of the sentences are calculated and used to generate the corresponding losses. Then, a direction to improve all the losses at the same time is found, and a gradient update on the policy model's parameters is performed.

calculate all m rewards for each (prompt z, input x, generated text y) triplet and optimize them.

The pseudo-code for this approach is provided in Figure 4, where at each update step, we start by sampling prompts from the policy model and use them to generate output sentences from another language model. We then calculate the reward values for each of the objectives separately and compute their corresponding losses. Then, a direction for improving all the losses simultaneously is calculated and used for the policy update. More details are available in Appendix §A.1.

4 Experiments

We now describe the empirical comparison of the RL-adapted multi-objective optimization methods that we introduced in the previous section. Our primary aim is to evaluate these techniques for discrete prompt optimization for downstream generative NLP tasks. Based on the availability of benchmarks and evaluation metrics, we focus on style transfer and machine translation tasks.

4.1 Tasks & Datasets

In this section, we describe the tasks, datasets, and their corresponding competing objectives. We evaluate on two tasks: unsupervised text style transfer and supervised machine translation.

We consider hypothetical tasks such as conveying positive sentiment as a competing objective in addition to accurate style transfer or machine translation. The selection of these specific objectives and tasks is motivated by the availability of standard evaluation datasets and well-established metrics within the NLP community. For style transfer, we focus on a specific sub-task that is well-supported by available ground-truth parallel style transfer data. Specifically, we aim to transfer modern English into a Shakespearean style. This particular style transfer task has long been a mainstay benchmark for the text style transfer NLP community (He et al., 2020; Deng et al., 2022).

Unsupervised Text Style Transfer. We experiment on the style transfer task (Xu et al., 2012; Jin et al., 2022), converting standard English into Shakespearean style. We consider three competing objectives: maintaining the original content of the input text, infusing it with Shakespearean style, and ensuring the resulting text conveys a positive sentiment. We test on the Shakespeare dataset (Xu et al., 2012; Jhamtani et al., 2017), and the objective function corresponding to content preservation is BertScore (Zhang et al., 2020), for sentiment is a sentiment RoBERTa-base classifier[2], and for style is a DistilBERT-base-uncased model fine-tuned on Shakespearean data[3].

It is noteworthy that while Shakespearean

[2] cardiffnlp/twitter-roberta-base-sentiment-latest
[3] notaphoenix/shakespeare_classifier_model

Method	Obj 1	Obj 2	Obj 3	Product	Average
Text Style Transfer (Obj_1: Content - Obj_2: Style - Obj_3: Sentiment)					
Average	19.56	**79.25**	38.28	30.91	**45.69**
Product	**34.58**	57.78	35.11	**36.04**	42.49
HVI	25.39	67.91	**38.76**	32.44	44.02
MGDA	22.37	66.51	38.11	31.16	42.33
Machine Translation (Obj_1: Content - Obj_2: BLEU - Obj_3: Sentiment)					
Average	32.07	**32.00**	46.36	65.48	36.81
Product	**32.95**	31.70	46.47	**65.98**	**37.04**
HVI	31.18	30.51	**48.69**	63.21	36.79
MGDA	31.46	31.85	46.03	62.87	36.45

Table 1: Reward values corresponding to each objective at a checkpoint where each method achieved the highest average of the product of rewards across samples. Even though the method utilizing the average of rewards achieved the highest average value for style transfer, we can observe an imbalance across various objective values. The product method, on the other hand, got the highest expected product value, reflecting a more balanced improvement. All the reported values are average objective values computed from 128 output samples.

style and positive sentiment may not directly be in conflict, they are not correlated either. Shakespeare's work includes many tragedies, such as "Hamlet," (Shakespeare, 1703) "Macbeth," (Shakespeare, 1710) etc., often with negative sentiments. On the other hand, Shakespeare has Comedies like "A Midsummer Night's Dream" (Shakespeare, 1734) that are more positive in terms of sentiment. Further, operationally, the way the objectives can conflict is that there may be word changes that easily improve sentiment reward (e.g., "AWESOME") that break the style reward and vice versa.

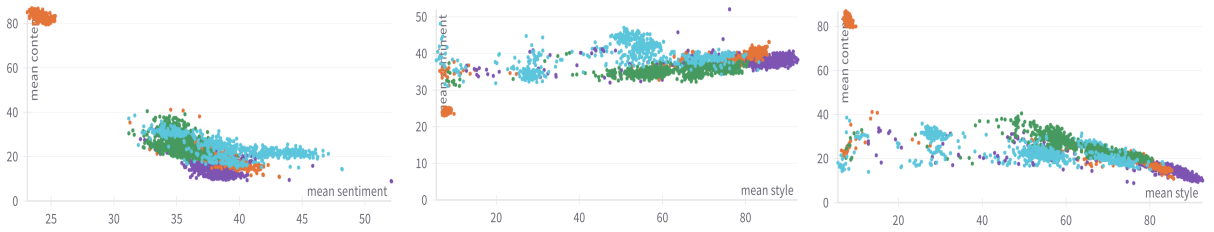
Supervised Machine Translation. We experiment on German to English translation task, using the *iwslt2017* data (Cettolo et al., 2017). The objectives and the reward functions are: (1) semantic similarity between the generated translation and a reference text computed using BertScore, (2) BLEU score (Papineni et al., 2002) between generated text and reference, and (3) conveying a positive sentiment quantified by the same RoBERTA-base classifier used in style-transfer task.

Evaluation Metrics We evaluate each task using its corresponding objective functions, with the goal of optimizing all rewards in a balanced manner. To quantify this balance, we assess performance by calculating both the mean and the expected product of the individual objectives for each task.

4.2 Training Details

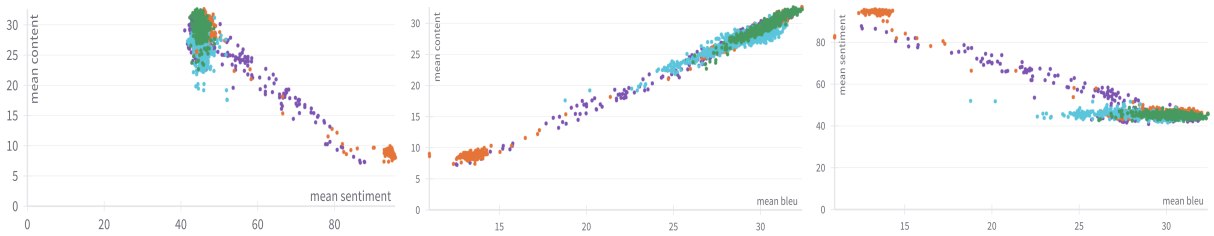
Following (Deng et al., 2022), we consider a multi-layer perception module on top of a small frozen distilGPT-2 model (Sanh et al., 2019), alongside a frozen LM head as the policy network. We employ an MLP with 3.1 million parameters. For the text style transfer task, we use a learning rate of 5e-5, while for the translation task, we set the learning rate to 1e-4. In both cases, we utilize the Adam optimizer. The policy network is trained for 12,000 steps. The number of training samples used for text style transfer and machine translation are 100 and 200, respectively. At each step, we sample eight prompts for a given input from the policy network, each comprising five tokens. Subsequently, we feed each prompt along with its corresponding input text into a separate LM to generate 128 output samples. We use GPT-2 (Radford et al., 2019) for text style transfer and flan-T5-small (Chung et al., 2022) for machine translation tasks.

Our choice of models was informed by an assessment of their respective strengths and capabilities in specific tasks. For instance, we observed that the flan-T5-small model exhibited superior performance in machine translation tasks compared to the GPT-2 model (Haddow et al., 2022); we followed past work in using the base models that tended to have a reasonable starting performance on the respective tasks. For instance, T5 has been repeatedly shown to be effective at translation tasks, while GPT-2



Sentiment vs Content | Style vs Sentiment | Style vs Content

Figure 5: Text Style Transfer. From left to right, positive sentiment vs. content match, Shakespearean style vs. positive sentiment, and Shakespearean style vs. content match for different settings of average reward, hyper volume indicator reward, expected product reward, and multiple gradient descent algorithm are shown.



Sentiment vs Content | BLEU vs Content | BLEU vs Sentiment

Figure 6: Pairwise reward values for Machine Translation Task from German to English, in different settings of average reward, hyper volume indicator reward, expected product reward, and multiple gradient descent algorithm.

fails to produce translations reliably. Further, we wanted to demonstrate that multi-objective optimization approaches could generalize across both the encoder-decoder and decoder-only language models. Moreover, there is a specific reason why we chose a weaker model like GPT-2 as it provides a better benchmark for multi-objective optimization precisely because GPT-2 is a weaker style transfer model out of the box compared to more recent models. As a result, there is a higher burden placed on the discrete prompt itself in order to achieve good results.

Furthermore, we employ "Efficient (soft) Q-learning" (Guo et al., 2022) to learn the policy network's parameters based on the reward using gradient descent.

We repeat each experiment with three distinct random seeds and report the average results. Using NVIDIA RTX A6000, each experiment takes about 20-24 hours.

4.3 Compared Methods

We compare two volume maximization approaches such as Hyper-volume indicator (HVI) and Expected product of rewards (Product). We also analyze the Multiple Gradient Descent Algorithm (MGDA) that finds the optimal gradient update direction to maximize all rewards simultaneously.

As a baseline, we use RLPrompt (Deng et al., 2022) for three objectives by optimizing the average of all rewards (Average) and comparing with the above-mentioned approaches.

5 Results

We report individual objective values as well as their expected product and average in Table 1. When evaluating based on the expected product, the product method demonstrates superior performance compared to the other approaches. Additionally, we observe a more balanced improvement across all rewards with volume-based methods such as HVI and product, in contrast to Average and MGDA. For example, in style transfer task, the "average" method improves *style* disproportionately higher than other objectives despite achieving the best performance based on the Average metric.

Among the compared methods, we believe that expected product of reward best captures the balanced performance across the rewards, as it mitigates the problems of *objective collapse* and *dominant outlier effect*, described in prior sections.

Method	Prompt	Input	Output	Obj 1	Obj 2	Obj 3
Text Style Transfer (Obj_1: Content - Obj_2: Style - Obj_3: Sentiment)						
HVI	thou WhereasYe WhereasOY	Be brave like the lion and proud.	Be thou like the lion and proud.	77.4	98.14	80.06
	EthiopJVphabet wherein whereby	He has to pay the price for his foolish actions.	He hath the price for his folly,	65.2	98.48	50.26
MGDA	Whereas ye untoWhereasð	Don't think I'm in love with him just because I'm asking about him.	Thou shalt not think that I'm in love with him just because I'm asking about him.	81.5	89.81	51.96
	SyriJV SHALL unto Ye	Take your penalty.	Take thy penalty.	98.03	44.14	83.3
Product	whereinWhereasWhereas whereinYe	He has to pay the price for his foolish actions.	He hath to pay the price for his folly;	81.1	98.36	47.07
	SHALL SHALL hath hath SHALL	Yes, my lord.	And shall be, my lord.	55.9	97.46	84.5

Table 2: Given the prompt learned from the policy model alongside the input to GPT2, the Shakespearian form of the sentence is generated as the output. The objective values corresponding to the output, as well as the method used for training the policy model, are reported.

5.1 Pairwise Reward Analysis

We plot the pairwise objective values achieved by each of the optimization methods on the validation set for text-style transfer and machine translation tasks in Figure 5 and 6 respectively. Each data point on the scatter plot represents the average objective value computed from 128 output samples, where each output sample is generated from a prompt sampled from the policy network and an input sentence from the validation dataset. Figure 5 illustrates how relying on the average of reward values can result in sacrifice of individual objectives in favor of overall improvement. We observe instances where sentiment and style scores are notably low, despite a high content score. This phenomenon arises due to the emphasis placed solely on the average of rewards, without consideration for individual objectives. MGDA performs slightly better than the average reward when balancing the individual objectives. However, the HVI and the product of rewards improve all the objectives simultaneously, with greater success.

Similarly, in the case of the machine translation task in Figure 6, we observe *objective collapse* for the average reward setting, while the other three approaches demonstrate a better balance among objectives while enhancing the joint reward. Notably, the HVI approach and the expected product of rewards are more successful in simultaneously optimizing all the objectives.

6 Qualitative Analysis

In this section, we perform a qualitative analysis of the learned prompts. In Table 2, we provide two examples of the style transfer task for each method, the generated prompt using it, the input, and the resulting output produced by the frozen model. The examples in Table 2, are some of the

successful examples chosen based on their high objective scores. In these examples, content is preserved reasonably well, while some of the words are changed in order to be more aligned with the Shakespearean style. The scores corresponding to the positive sentiment scores should be interpreted carefully, as achieving a more positive sentiment might change the semantic meaning of the sentence to some extent, specifically where the original input has an opposite sentiment. For instance, in another example sentence, "crimes" was replaced with "good deeds" to make the sentiment more positive, at the expense of getting a lower score on the competing objective of content preservation. Overall, the models seem to achieve a balanced performance when handling these conflicting situations.

Moreover, we can observe some similarities and common words in high-performing prompts, demonstrating the effectiveness of certain tokens for a specific task. However, we can see that despite these similarities, the differences in the prompts from various methods can substantially affect the final evaluation results, which were shown in Table 1.

It is important to note that the highest-performing prompts do not necessarily need to be interpretable by humans. There is value in a discrete prompt beyond it being interpretable by humans: a discrete prompt, unlike a continuous prompt, can be passed into a black-box model like ChatGPT, while a continuous prompt cannot. Furthermore, discrete prompts are more likely to generalize across different LLMs than continuous prompts due to the common text space instead of the model-specific latent space (Deng et al., 2022). In applications where having an interpretable prompt is useful, corresponding rewards could be included in the multi-objective formulation.

7 Related Work

Prompt Tuning. A line of research has emerged with a focus on improving the discrete (Jiang et al., 2020; Prasad et al., 2023; Mishra et al., 2022) and soft prompts (Li and Liang, 2021; Qin and Eisner, 2021; Vu et al., 2022; Liu et al., 2023) for improved downstream performance. Few recent works generate discrete prompts by utilizing the models gradients (Shin et al., 2020; Wen et al., 2023), employing evolution algorithms (Guo et al., 2023), and reinforcement learning (Zhang et al., 2023; Deng et al., 2022; Jung and Kim, 2023; Wang et al., 2023). Our work shares a similar direction, but we focus on multiple competing objectives instead of one.

Multi-objective Reinforcement Learning. Multi-objective reinforcement learning is typically studied in decision-making (Van Moffaert et al., 2013; Van Moffaert and Nowé, 2014; Yang et al., 2019; Xu et al., 2020; Hayes et al., 2022). Jang et al. (2023) fine-tunes LMs for multiple objectives by training one policy model per objective and merging them. (Lin et al., 2019; Sener and Koltun, 2019) perform multi-objective RL in a multi-task learning setup. Instead, we propose optimizing the prompts for one model with multiple objectives.

8 Conclusion

We empirically investigate the use of optimization techniques alongside reinforcement learning to address discrete prompt optimization in a multi-objective context. Our experiments show that multi-objective methods, which directly optimize the volume, outperform those seeking monotonic update directions, achieving a better balance across all rewards.

9 Limitations

The methods discussed in this paper take many GPU hours to converge, making it computationally expensive to run. Moreover, our optimization methods perform well on smaller LMs like GPT2, we have not experimented with larger models because of the substantial computational cost.

10 Ethical Considerations

This paper introduces three approaches for discrete prompt optimization. As such, prompt-tuning should not introduce biases not already observed in the model and generate any harmful text as prompts, and we do not anticipate any significant ethical concerns.

References

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jörg Fliege and Benar Fux Svaiter. 2000. Steepest descent methods for multicriteria optimization. *Mathematical methods of operations research*, 51:479–494.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. 2022. Efficient (soft) q-learning for text generation with limited good data.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Barry Haddow, Rachel Bawden, Antonio Valerio Miceli Barone, Jindřich Helcl, and Alexandra Birch. 2022. Survey of low-resource machine translation. *Computational Linguistics*, 48(3):673–732.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. *ArXiv*, abs/2002.03912.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19, Copenhagen, Denmark. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2022. Deep learning for text style transfer: A survey. *Computational Linguistics*, 48(1):155–205.

Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete prompt compression with reinforcement learning. *arXiv preprint arXiv:2308.08758*.

Joshua Knowles, David Corne, and Mark Fleischer. 2004. Bounded archiving using the lebesgue measure.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. 2019. Pareto multi-task learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022. Reframing instructional prompts to GPTk's language. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 589–612, Dublin, Ireland. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association*

for Computational Linguistics, pages 3845–3864, Dubrovnik, Croatia. Association for Computational Linguistics.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Ozan Sener and Vladlen Koltun. 2019. Multi-task learning as multi-objective optimization.

William Shakespeare. 1703. *The tragedy of Hamlet, prince of Denmark*. Wellington.

William Shakespeare. 1710. *Macbeth. A tragedy. With all the alterations, amendments, additions, and new songs [by Sir William Davenant], as it is now acted at the Queen's Theatre.[Anonymous.]*. J. Tonson.

William Shakespeare. 1734. *A Midsummer Night's Dream*, volume 2. J. TONSON, and the rest of the PROPRIETORS; and sold.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. 2013. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, pages 191–199. IEEE.

Kristof Van Moffaert and Ann Nowé. 2014. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. 2023. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *arXiv preprint arXiv:2302.03668*.

Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*, pages 10607–10616. PMLR.

Wei Xu, Alan Ritter, Bill Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914, Mumbai, India. The COLING 2012 Organizing Committee.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert.

Eckart Zitzler and Lothar Thiele. 1998. Multiobjective optimization using evolutionary algorithms — a comparative case study. In *Parallel Problem Solving from Nature — PPSN V*, pages 292–301, Berlin, Heidelberg. Springer Berlin Heidelberg.

A Appendix

A.1 Multiple Gradient Descent Algorithm

(Fliege and Svaiter, 2000) proposes a steepest descent algorithm for multi-criteria optimization, where the update rule for the parameters θ at time t with the step size η is defined as:

$$\theta_{t+1} = \theta_t - \eta d_t \tag{4}$$

where the search direction d_t is calculated as follows, with $\mathcal{L}_i(\theta_j)$ being the expected loss corresponding to objective o_i:

$$\begin{aligned} &(d_t, \alpha_t) = \arg \min_{d \in \mathbb{R}^n, \alpha \in \mathbb{R}} \alpha + \frac{1}{2}\|d\|^2, \\ &\text{s.t. } \nabla \mathcal{L}_i \left(\theta_t\right)^T d \leq \alpha, \quad i = 1, \ldots, m. \end{aligned} \tag{5}$$

A valid direction d_t improves the values for all the objectives, simultaneously. Moreover, (Fliege and Svaiter, 2000) shows that the solution obtained by the aforementioned approach leads to a Pareto critical point.

Based on the KKT conditions, we have

$$d_t = -\left(\sum_{i=1}^{m} \lambda_i \nabla \mathcal{L}_i \left(\theta_t\right)\right), \quad \sum_{i=1}^{m} \lambda_i = 1 \tag{6}$$

and we can write equation-5 in its dual form:

$$\begin{aligned} &\max_{\lambda_i} -\frac{1}{2}\left\|\sum_{i=1}^{m} \lambda_i \nabla \mathcal{L}_i \left(\theta_t\right)\right\|^2 \\ &\text{s.t. } \sum_{i=1}^{m} \lambda_i = 1, \lambda_i \geq 0, , \forall i = 1, \ldots, m. \end{aligned} \tag{7}$$

Therefore, in order to find a valid direction d that improves all the objectives, we can rewrite the equation in its dual form. This will give us a constrained optimization problem, which can be solved by the Frank-Wolfe algorithm, and we can then use gradient descent to update the policy parameters.